Drawer

Drawer The <ava-drawer> component is a powerful sliding panel that provides a flexible overlay interface for navigation, forms, content display, and interactive elements. It supports multiple positions, sizes, animations, and advanced features like resizing and persistence, making it ideal for creating modern, accessible user interfaces. How to use ■ import { AavaDrawerComponent } from "@aava/play-core" ; Basic Usage ■ A simple drawer with default right position and medium size. Positions ■ The drawer component supports four different positions for various use cases. Sizes ■ Choose from five predefined sizes or use custom dimensions. Content Structure ■ Organize drawer content with header, body, and footer sections. Features ■ Multiple Positions ■ Right : Default position, slides in from the right edge Left : Slides in from the left edge, perfect for navigation Top : Slides down from the top, ideal for notifications Bottom : Slides up from the bottom, great for forms Size Variants ■ Small : 320px width (200px height for top/bottom) Medium : 480px width (300px height for top/bottom) - Default Large : 640px width (400px height for top/bottom) Extra Large : 800px width (500px height for top/bottom) Full : 100% viewport width/height Custom : Specify exact dimensions with width/height properties Advanced Features ■ Resizable : Enable drag-to-resize functionality Persistent : Prevent accidental closing Custom Animations : Spring-based animations with reduced motion support Overlay Control : Configurable overlay behavior Keyboard Navigation : Full keyboard accessibility Focus Management : Proper focus trapping and restoration Accessibility ■ ARIA Support : Complete ARIA attributes and roles Screen Reader : Full screen reader compatibility Keyboard Navigation : Escape key, tab trapping Reduced Motion : Respects user motion preferences High Contrast : High contrast mode support API Reference ■ Inputs ■ Property Type Default Description isOpen boolean false Controls the visibility of the drawer position DrawerPosition 'right' Position of the drawer on screen size DrawerSize 'medium' Predefined size of the drawer showOverlay boolean true Whether to show the backdrop overlay closeOnOverlayClick boolean true Close drawer when overlay is clicked closeOnEscape boolean true Close drawer when Escape key is pressed showCloseButton boolean true Show the close button in header persistent boolean false Prevent drawer from being closed resizable boolean false Enable resize functionality animate boolean true Enable animations title string '' Title displayed in the header subtitle string '' Subtitle displayed in the header showHeader boolean true Show the header section showFooter boolean false Show the footer section width string '' Custom width (overrides size) height string '' Custom height (overrides size) maxWidth string '' Maximum width constraint maxHeight string '' Maximum height constraint zIndex number 1050 Z-index for drawer positioning closeIcon string 'X' Icon name for close button closeIconSize number 20 Size of the close icon Outputs ■ Event Type Description opened EventEmitter<void> Emitted when drawer opens closed EventEmitter<void> Emitted when drawer closes overlayClick EventEmitter<void> Emitted when overlay is clicked escapePressed EventEmitter<void> Emitted when Escape key is pressed Methods ■ Method Parameters Return Type Description open() None void Opens the drawer close() None void Closes the drawer

toggle() None void Toggles drawer open/closed state onOverlayClick() None void Handles overlay click event onCloseClick() None void Handles close button click onDrawerClick() Event void Prevents event bubbling getDrawerClasses() None string Returns CSS classes for drawer getOverlayClasses() None string Returns CSS classes for overlay getDrawerStyles() None object Returns inline styles for drawer Types ■ DrawerPosition ■ type DrawerPosition = "left" | "right" | "top" | "bottom" ; DrawerSize ■ type DrawerSize = "small" | "medium" | "large" | "extra-large" | "full" ; Content Projection ■ The drawer component supports content projection with specific slots: Slot Name Description [slot=header] Custom content for the header section [slot=footer] Custom content for the footer section Default Main content displayed in the body section CSS Classes ■ The component provides several CSS classes for styling: Class Name Description .ava-drawer Main drawer container .ava-drawer--open Applied when drawer is open .ava-drawer--left Left position styling .ava-drawer--right Right position styling .ava-drawer--top Top position styling .ava-drawer--bottom Bottom position styling .ava-drawer--small Small size styling .ava-drawer--medium Medium size styling .ava-drawer--large Large size styling .ava-drawer--extra-large Extra large size styling .ava-drawer--full Full size styling .ava-drawer--resizable Resizable functionality styling .ava-drawer-overlay Overlay container .ava-drawer-overlay--open Applied when overlay is visible .ava-drawer__animation-wrapper Animation wrapper .ava-drawer__content Content container .ava-drawer__header Header section .ava-drawer__header-content Header content area .ava-drawer__title-section Title and subtitle container .ava-drawer__title Title element .ava-drawer__subtitle Subtitle element .ava-drawer__header-slot Header slot container .ava-drawer__close-section Close button container .ava-drawer__body Main content area .ava-drawer__footer Footer section .ava-drawer__resize-handle Resize handle CSS Custom Properties ■ The component uses CSS custom properties for theming: Property Description --drawer-background Background color of the drawer --drawer-border Border styling for the drawer --drawer-shadow Box shadow for the drawer --drawer-z-index Z-index for drawer positioning --drawer-spring-duration Animation duration (300ms) --drawer-spring-easing Animation easing function --drawer-header-padding Header padding --drawer-header-border Header bottom border --drawer-header-background Header background color --drawer-header-gap Gap between header elements --drawer-title-font Title font styling --drawer-title-color Title text color --drawer-title-weight Title font weight --drawer-title-line-height Title line height --drawer-subtitle-font Subtitle font styling --drawer-subtitle-color Subtitle text color --drawer-subtitle-weight Subtitle font weight --drawer-subtitle-line-height Subtitle line height --drawer-subtitle-margin Subtitle margin --drawer-body-padding Body padding --drawer-body-background Body background color --drawer-body-text-color Body text color --drawer-footer-padding Footer padding --drawer-footer-border Footer top border --drawer-footer-background Footer background color --popup-overlay-background Overlay background color --global-motion-duration-standard Standard motion duration --global-motion-easing-standard Standard motion easing --global-spacing-2 Small spacing token --global-spacing-3 Medium spacing token Best Practices ■ User Experience ■ Appropriate Size : Choose drawer size based on content complexity Clear

Purpose : Use descriptive titles and content structure Smooth Interactions : Enable animations for better UX Consistent Behavior : Maintain consistent drawer patterns across the app Responsive Design : Ensure drawers work well on all screen sizes Performance ■ Conditional Rendering : Only render drawer content when needed Animation Optimization : Use hardware-accelerated animations Memory Management : Clean up event listeners and timeouts Change Detection : Leverage OnPush strategy for better performance Content Loading : Load heavy content after drawer opens Accessibility ■ Keyboard Navigation : Ensure all interactions work with keyboard Screen Reader Support : Provide proper ARIA labels and descriptions Focus Management : Trap focus within drawer when open Motion Preferences : Respect user's reduced motion preferences High Contrast : Ensure visibility in high contrast mode Content Organization ■ Header Structure : Use clear titles and optional subtitles Content Hierarchy : Organize content logically within body Footer Actions : Place primary actions in footer when appropriate Scrolling : Ensure long content scrolls properly within body Responsive Content : Adapt content layout for different drawer sizes Accessibility Guidelines ■ Screen Reader Support ■ ARIA Attributes : Proper role="dialog" , aria-modal="true" Labels : aria-labelledby and aria-describedby for content identification State Announcements : Clear announcements for drawer state changes Content Structure : Semantic HTML structure for better navigation Keyboard Navigation ■ Tab Order : Logical tab order within the drawer Escape Key : Close drawer with Escape key (configurable) Focus Trapping : Focus remains within drawer when open Focus Restoration : Return focus to trigger element when closed Visual Design ■ High Contrast : Ensure sufficient contrast ratios Focus Indicators : Clear focus indicators for all interactive elements Color Independence : Don't rely solely on color for information Text Scaling : Support for text scaling and zoom Motion and Animation ■ Reduced Motion : Respect prefers-reduced-motion media query Animation Duration : Keep animations smooth but not distracting Motion Alternatives : Provide alternatives for users who can't see animations Performance : Ensure animations don't cause performance issues Responsive Behavior ■ Mobile Adaptations ■ The drawer component automatically adapts to mobile screens: Full Width/Height : Drawers become full viewport on mobile Touch Optimization : Optimized for touch interactions Viewport Adaptation : Adapts to different mobile viewport sizes Performance : Optimized performance for mobile devices Breakpoint Behavior ■ Desktop (>768px) : Full drawer with all features Mobile (≤768px) : Optimized drawer for mobile screens Content Scaling : Drawer content scales appropriately Animation Performance : Optimized animations for different devices Content Considerations ■ Flexible Sizing : Drawer adapts to different content sizes Layout Preservation : Maintains layout consistency across devices Loading States : Consistent loading experience across platforms Performance : Efficient rendering on all device types

```
<aava-drawer
  [isOpen]="isDrawerOpen"
  title="Basic Drawer"
  subtitle="This is a simple drawer example"
  (opened)="onDrawerOpened()"
  (closed)="onDrawerClosed()"
  (closed)="closeDrawer()"
>
  <div class="drawer-content">
    <h4>Welcome to the Drawer!</h4>
    <p>This is a basic drawer component with default settings:</p>
    <ul>
      <li>Position: Right (default)</li>
      <li>Size: Medium (default)</li>
      <li>Overlay: Enabled</li>
      <li>Animations: Enabled</li>
    </ul>
    <p>You can close this drawer by:</p>
    <ul>
      <li>Clicking the X button</li>
      <li>Clicking the overlay</li>
      <li>Pressing the Escape key</li>
    </ul>
  </div>

  <div slot="footer">
    <aava-button label="Close" variant="secondary" (click)="closeDrawer()">
    </aava-button>
    <aava-button label="Save" variant="primary" (click)="closeDrawer()">
    </aava-button>
  </div>
</aava-drawer>


---

  isDrawerOpen = false;

  openDrawer(): void {
    this.isDrawerOpen = true;
  }

  closeDrawer(): void {
    this.isDrawerOpen = false;
  }

  onDrawerOpened(): void {
    console.log('Drawer opened');
  }

  onDrawerClosed(): void {
    console.log('Drawer closed');
  }
```

```
<aava-drawer
  [isOpen]="rightDrawerOpen"
  position="right"
  title="Right Drawer"
  (closed)="closeDrawer('right')"
>
  <div class="drawer-content">
    <h4>Right Position</h4>
    <p>This drawer slides in from the right side.</p>
    <p>This is the default position for drawers.</p>
    <ul>
      <li>Perfect for forms and detail panels</li>
      <li>Common pattern for editing content</li>
      <li>Good for mobile-first designs</li>
    </ul>
  </div>
</aava-drawer>

<!-- Left Position -->
<aava-drawer
  [isOpen]="leftDrawerOpen"
  position="left"
  title="Left Drawer"
  (closed)="closeDrawer('left')"
>
  <div class="drawer-content">
    <h4>Left Position</h4>
    <p>This drawer slides in from the left side.</p>
    <p>Perfect for navigation menus and sidebars.</p>
    <ul>
      <li>Traditional navigation pattern</li>
      <li>Good for hierarchical menus</li>
      <li>Familiar to users</li>
    </ul>
  </div>
</aava-drawer>

<!-- Top Position -->
<aava-drawer
  [isOpen]="topDrawerOpen"
  position="top"
  title="Top Drawer"
  (closed)="closeDrawer('top')"
>
  <div class="drawer-content">
    <h4>Top Position</h4>
    <p>This drawer slides in from the top.</p>
    <p>Great for notifications or quick actions.</p>
    <ul>
      <li>Perfect for notifications</li>
      <li>Good for search interfaces</li>
      <li>Quick access to tools</li>
    </ul>
  </div>
</aava-drawer>
```

```
<!-- Bottom Position -->
<aava-drawer
  [isOpen]="bottomDrawerOpen"
  position="bottom"
  title="Bottom Drawer"
  (closed)="closeDrawer('bottom')"
>
  <div class="drawer-content">
    <h4>Bottom Position</h4>
    <p>This drawer slides in from the bottom.</p>
    <p>Ideal for mobile-first designs and action sheets.</p>
    <ul>
      <li>Mobile-friendly pattern</li>
      <li>Good for action sheets</li>
      <li>Thumb-friendly interaction</li>
    </ul>
  </div>
</aava-drawer>


---

 // Drawer states for different positions
  rightDrawerOpen = false;
  leftDrawerOpen = false;
  topDrawerOpen = false;
  bottomDrawerOpen = false;

  openDrawer(position: string): void {
    switch (position) {
      case 'right':
        this.rightDrawerOpen = true;
        break;
      case 'left':
        this.leftDrawerOpen = true;
        break;
      case 'top':
        this.topDrawerOpen = true;
        break;
      case 'bottom':
        this.bottomDrawerOpen = true;
        break;
    }
  }

  closeDrawer(position: string): void {
    switch (position) {
      case 'right':
        this.rightDrawerOpen = false;
        break;
      case 'left':
        this.leftDrawerOpen = false;
        break;
      case 'top':
        this.topDrawerOpen = false;
        break;
```

```
      case 'bottom':
        this.bottomDrawerOpen = false;
        break;
    }
  }
```

■ No code found

■ No code found